An Overview of Trilinos

Jonathan Hu
Sandia National Laboratories

Tenth DOE ACTS Collection Workshop
August 20th, 2009

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.

Outline of Talk

- Background / Motivation / Evolution.
- Trilinos Package Concepts.
- Whirlwind Tour of Trilinos Packages.
- Getting Started.
- Concluding remarks.

- Hands On Tutorial

Trilinos Development Team

Chris Baker
Developer of Anasazi, RBGen, Tpetra

Ross Bartlett
Lead Developer of Thyra and Stratimikos
Developer of Rythmos

Pavel Bochev
Project Lead and Developer of Intrepid

Paul Boggs
Developer of Thyra

Eric Boman
Lead Developer of Isorropia
Developer of Zoltan

Todd Coffey
Lead Developer of Rythmos

David Day
Developer of Komplex and Intrepid

Karen Devine
Lead Developer of Zoltan

Clark Dohrmann
Developer of CLAPS

Michael Gee
Developer of ML, NOX

Bob Heaphy
Lead Developer of Trilinos SQA

Mike Heroux
Trilinos Project Leader
Lead Developer of Epetra, AztecOO,
Kokkos, Komplex, IFPACK, Thyra, Tpetra
Developer of Amesos, Belos, EpetraExt, Jpetra

Ulrich Hetmaniuk
Developer of Anasazi

Robert Hoekstra
Lead Developer of EpetraExt
Developer of Epetra, Thyra, Tpetra

Russell Hooper
Developer of NOX

Vicki Howle
Lead Developer of Meros
Developer of Belos and Thyra

Jonathan Hu
Developer of ML

Sarah Knepper
Developer of Komplex

Tammy Kolda
Lead Developer of NOX

Joe Kotulski
Lead Developer of Pliris

Rich Lehoucq
Developer of Anasazi and Belos

Kevin Long
Lead Developer of Thyra, Sundance
Developer of Teuchos

Roger Pawlowski
Lead Developer of NOX, Phalanx
Developer of Shards, LOCA

Michael Phenow
Trilinos Webmaster
Lead Developer of New_Package

Eric Phipps
Lead Developer of Sacado
Developer of LOCA, NOX

Denis Ridzal
Lead Developer of Aristos and Intrepid

Marzio Sala
Lead Developer of Didasko and IFPACK
Developer of ML, Amesos

Andrew Salinger
Lead Developer of LOCA

Paul Sexton
Developer of Epetra and Tpetra

Bill Spotz
Lead Developer of PyTrilinos
Developer of Epetra, New_Package

Ken Stanley
Lead Developer of Amesos and New_Package

Heidi Thornquist
Lead Developer of Anasazi, Belos, RBGen, and Teuchos

Ray Tuminaro
Lead Developer of ML and Meros

Jim Willenbring
Developer of Epetra and New_Package.
Trilinos library manager

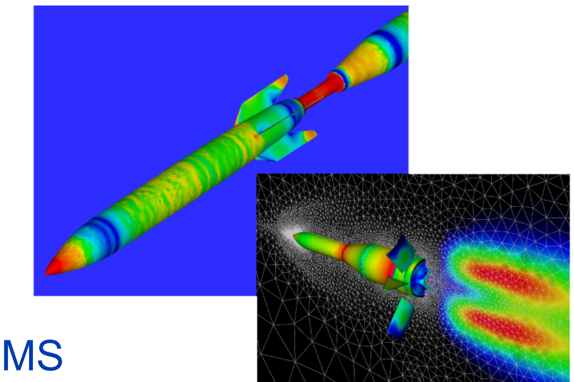
Alan Williams
Lead Developer of Isorropia
Developer of Epetra, EpetraExt, AztecOO, Tpetra

Sandia National Laboratories

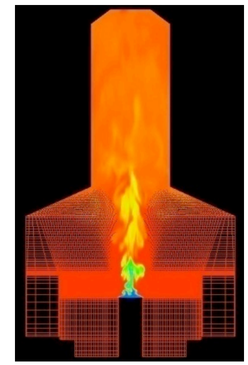
Sandia Physics Simulation Codes

- Element-based
  - Finite element, finite volume, finite difference, network, etc…
- Large-scale
  - Billions of unknowns
- Parallel
  - MPI-based SPMD
  - Distributed memory
- C++
  - Object oriented
  - Some coupling to legacy Fortran libraries

Fluids

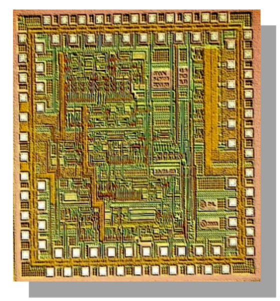


Combustion

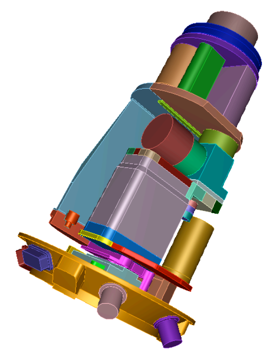


MEMS

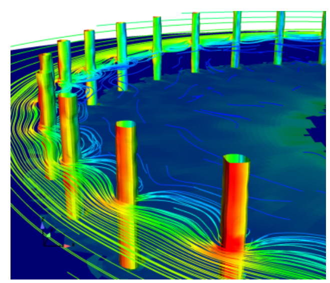
Circuits



Structures



Plasmas



Sandia National Laboratories

Motivation For Trilinos

- Sandia does LOTS of solver work.
- 9 years ago …
  - Aztec was a mature package. Used in many codes.
  - FETI, PETSc, DSCPack, Spooles, ARPACK, DASPK, and many other codes were (and are) in use.
  - New projects were underway or planned in multi-level preconditioners, eigensolvers, non-linear solvers, etc…
- The challenges:
  - Little or no coordination was in place to:
    - Efficiently reuse existing solver technology.
    - Leverage new development across various projects.
    - Support solver software processes.
    - Provide consistent solver APIs for applications.
  - ASCI was forming software quality assurance/engineering (SQA/SQE) requirements:
    - Daunting requirements for any single solver effort to address alone.

Sandia National Laboratories

Evolving Trilinos Solution

- Trilinos[1] is an evolving framework to address these challenges:
  - Follow a TOOLKIT approach.
  - Fundamental atomic unit is a *package*.
  - Includes core set of vector, graph and matrix classes (Epetra/Tpetra packages).
  - Provides a common abstract solver API (Thyra package).
  - Provides a ready-made package infrastructure (new_package package):
    - Source code management (cvs, bonsai).
    - Build tools (autotools for 9.0, cmake beginning with 10.0).
    - Automated regression testing (queue directories within repository).
    - Communication tools (mailman mail lists).
  - Specifies requirements and suggested practices for package SQA.
- In general allows us to categorize efforts:
  - Efforts best done at the Trilinos level (useful to most or all packages).
  - Efforts best done at a package level (peculiar or important to a package).
  - **Allows package developers to focus only on things that are unique to their package.**

1. Trilinos loose translation: "A string of pearls"

Evolving Trilinos Solution

- Beyond a “solvers” framework
- Natural expansion of capabilities to satisfy application and research needs

physics

$L(u)=f$
Math. model

$L_h(u_h)=f_h$
Numerical model

$u_h=L_h^{-1} \cdot f_h$
Algorithms

computation

Numerical math
Convert to models that can be solved on digital computers

Algorithms
Find faster and more efficient ways to solve numerical models

discretizations
Time domain
Space domain

methods
Automatic diff.
Domain dec.
Mortar methods

Trilinos

solvers
Linear
Nonlinear
Eigenvalues
Optimization

core
Petra
Utilities
Interfaces
Load Balancing

- Discretization methods, AD, Mortar methods, …

Characterizing the Trilinos "Project"

- Not a "project" but an infrastructure to support inter-related projects: A project of projects.
- Package participation is voluntary:
  - Framework must be attractive (and continue to be).
  - Requirements are few, opportunities are many.
  - Package team decides what and when.
  - Opt-out is always an option.
- Package autonomy is carefully guarded:
  - Even if redundant development occurs.
  - Decision-making pushed to lowest (best) level.
- Participation is attractive:
  - Increasing infrastructure capabilities.
  - Access to many other packages.

Trilinos Strategic Goals

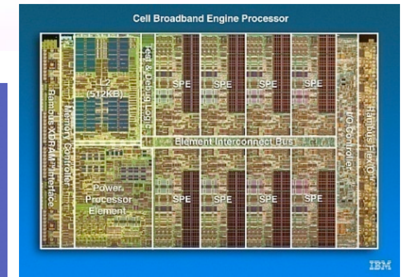
- **Scalable Computations:** As problem size and processor counts increase, the cost of the computation will remain nearly fixed.
- **Hardened Computations:** Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage:** Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis and optimization.

Algorithmic Goals

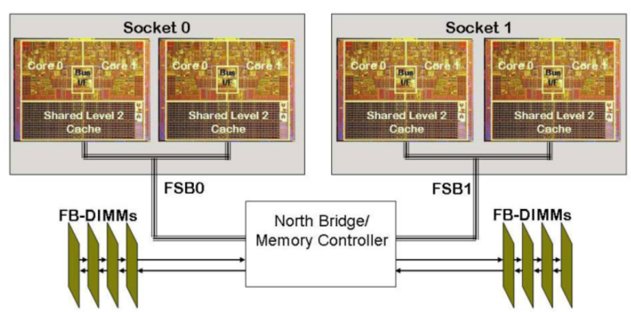
- ***Grand* Universal Interoperability**: All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Accessibility:** All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.
- **Universal Solver RAS**: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: Easy to maintain and upgrade within the application environment.

Software Goals

Target Platforms: Any and All
(Now and in the Future)



- Desktop: Development and more…
- Capability machines:
  - Redstorm (XT3), Clusters
  - Roadrunner (Cell-based).
  - Multicore nodes.
- Parallel software environments:
  - MPI of course.
  - UPC, CAF, threads, vectors,…
  - Combinations of the above.
- User "skins":
  - C++/C, Python
  - Fortran.
  - Web, CCA.



Trilinos Package Summary

http://trilinos.sandia.gov

	Objective	Package(s)
Discretizations	Meshing & Spatial Discretizations	phdMesh, Intrepid, Phalanx, Shards, Pamgen, Sundance
	Time Integration	Rythmos
Optimization	Optimization (SAND)	MOOCHO, Aristos
Methods	Automatic Differentiation	Sacado
	Mortar Methods	Moertel
Core	Linear algebra objects	Epetra, Jpetra, Tpetra
	Abstract interfaces	Thyra, Stratimikos, RTOp
	Load Balancing	Zoltan, Isorropia
	"Skins"	PyTrilinos, WebTrilinos, Star-P, ForTrilinos, CTrilinos
	C++ utilities, (some) I/O	Teuchos, EpetraExt, Kokkos, Triutils
Preconditioners	Multigrid methods	ML
	Domain decomposition methods	CLAPS, IFPACK
	ILU-type methods	AztecOO, IFPACK
	Block preconditioners	Meros
Solvers	Iterative (Krylov) linear solvers	AztecOO, Belos, Komplex
	Direct sparse linear solvers	Amesos
	Direct dense linear solvers	Epetra, Teuchos, Pliris
	Nonlinear system solvers	NOX, LOCA
	Iterative eigenvalue solvers	Anasazi
	Stochastic PDEs	Stokhos

Trilinos

Package Concepts

Interoperability vs. Dependence

("Can Use") ("Depends On")

- Although most Trilinos packages have no explicit dependence, often packages must interact with *some* other packages:
  - NOX needs operator, vector and linear solver objects.
  - AztecOO needs preconditioner, matrix, operator and vector objects.
  - Interoperability is enabled at configure time. For example, NOX:

`--enable-nox-lapack`	compile NOX lapack interface libraries
`--enable-nox-epetra`	compile NOX epetra interface libraries
`--enable-nox-petsc`	compile NOX petsc interface libraries

- Trilinos **`configure`** script is vehicle for:
  - Establishing interoperability of Trilinos components…
  - Without compromising individual package autonomy.
- Trilinos offers seven basic interoperability mechanisms.

Trilinos Interoperability Mechanisms

(Acquired as Package Matures)

Package builds under Trilinos configure scripts.	⇒	*Package* can be built as part of a suite of packages; cross-package interfaces enable/disable automatically
Package accepts user data as Epetra or Thyra objects	⇒	Applications using Epetra/Thyra can use *package*
Package accepts parameters from Teuchos ParameterLists	⇒	Applications using Teuchos ParameterLists can drive *package*
Package can be used via Thyra abstract solver classes	⇒	Applications or other packages using Thyra can use *package*
Package can use Epetra for private data.	⇒	*Package* can then use other packages that understand Epetra
Package accesses solver services via Thyra interfaces	⇒	*Package* can then use other packages that implement Thyra interfaces
Package available via PyTrilinos	⇒	*Package* can be used with other Trilinos packages via Python.

What Trilinos is not …

- Trilinos is not a single monolithic piece of software. Each package:
  - Can be built independent of Trilinos.
  - Has its own self-contained CVS structure.
  - Has its own Bugzilla product and mail lists.
  - Development team is free to make its own decisions about algorithms, coding style, release contents, testing process, etc.
- Trilinos top layer is not a large amount of source code:
  - Trilinos repository (6.0 branch) contains: 660,378 source lines of code (SLOC).
  - Sum of the packages SLOC counts: 648,993.
  - Trilinos top layer SLOC count: 11,385 (1.7%).
- Trilinos is not "indivisible":
  - You don't need all of Trilinos to get things done.
  - Any collection of packages can be combined and distributed.
  - Current public release contains only 26 of the 30+ Trilinos packages.

Whirlwind Tour of Packages

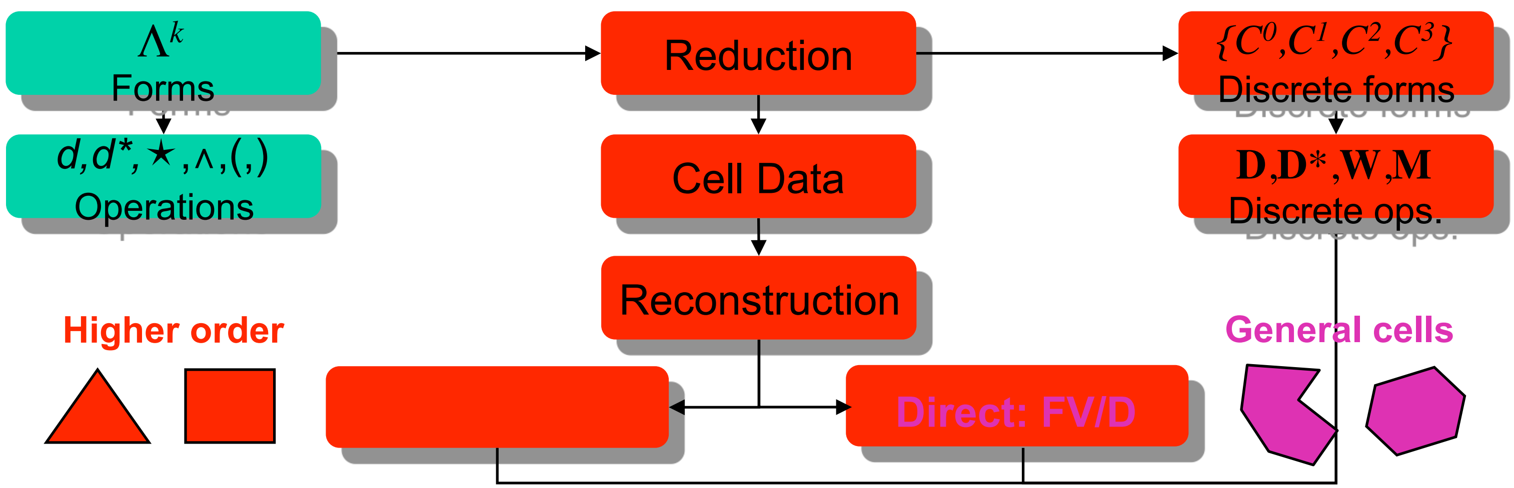
Discretizations Methods Core Solvers/Preconditioners

Intrepid

Interoperable Tools for Rapid Development of Compatible Discretizations

Intrepid offers an **innovative software design** for compatible discretizations:

- allows access to FEM, FV and FD methods using a common API
- supports hybrid discretizations (FEM, FV and FD) on unstructured grids
- supports a variety of cell shapes:
    - standard shapes (e.g. tets, hexes): high-order finite element methods
    - arbitrary (polyhedral) shapes: low-order mimetic finite difference methods
- enables optimization, error estimation, V&V, and UQ using fast invasive techniques (direct support for cell-based derivative computations or via automatic differentiation)



Developers: Pavel Bochev and Denis Ridzal

Rythmos

- Suite of time integration (discretization) methods

- Includes: backward Euler, forward Euler, explicit Runge-Kutta, and implicit BDF at this time.
- Native support for operator split methods.
- Highly modular.
- Forward sensitivity computations will be included in the first release with adjoint sensitivities coming in near future.

Developers: Todd Coffey, Roscoe Bartlett

Sandia National Laboratories

Whirlwind Tour of Packages

Discretizations **Methods** **Core** **Solvers/Preconditioners**

Sandia National Laboratories

Sacado: Automatic Differentiation

- Efficient OO based AD tools optimized for element-level computations

- Applies AD at “element”-level computation
  - “Element” means finite element, finite volume, network device,…
- Template application’s element-computation code
  - Developers only need to maintain one templated code base
- Provides three forms of AD
  - Forward Mode: $(x,\ V) \longrightarrow \left(f,\ \frac{\partial f}{\partial x}V\right)$
    - Propagate derivatives of intermediate variables w.r.t. independent variables forward
    - Directional derivatives, tangent vectors, square Jacobians, $\partial f/\partial x$ when **m ≥ n**.
  - Reverse Mode: $(x,\ W) \longrightarrow \left(f,\ W^T\frac{\partial f}{\partial x}\right)$
    - Propagate derivatives of dependent variables w.r.t. intermediate variables backwards
    - Gradients, Jacobian-transpose products (adjoints), $\partial f/\partial x$ when **n > m**.
  - Taylor polynomial mode: $x(t) = \sum_{k=0}^{d} x_k t^k \longrightarrow \sum_{k=0}^{d} f_k t^k = f(x(t)) + O(t^{d+1}), \quad f_k = \frac{1}{k!}\frac{d^k}{dt^k}f(x(t))$
  - Basic modes combined for higher derivatives.

Developers: Eric Phipps, David Gay

Sandia National Laboratories

Trilinos

Whirlwind Tour of Packages

Discretizations Methods Core Solvers/Preconditioners

Teuchos

- Portable utility package of commonly useful tools:
  - ParameterList class: key/value pair database, recursive capabilities.
  - LAPACK, BLAS wrappers (templated on ordinal and scalar type).
  - Dense matrix and vector classes (compatible with BLAS/LAPACK).
  - FLOP counters, timers.
  - Ordinal, Scalar Traits support: Definition of ‘zero’, ‘one’, etc.
  - Reference counted pointers / arrays, and more…
- Takes advantage of advanced features of C++:
  - Templates
  - Standard Template Library (STL)
- Teuchos::ParameterList:
  - Allows easy control of solver parameters.
  - XML format input/output.

Developers: Roscoe Barlett, Kevin Long, Heidi Thornquist, Mike Heroux, Paul Sexton, Kris Kampshoff, Chris Baker

Trilinos Common Language: Petra

- Petra provides a "common language" for distributed linear algebra objects (operator, matrix, vector)

- Petra[1] provides distributed matrix and vector services.
- Exists in basic form as an object model:
  - Describes basic user and support classes in UML, independent of language/implementation.
  - Describes objects and relationships to build and use matrices, vectors and graphs.
  - Has 3 implementations under development.

[1]Petra is Greek for "foundation".

Petra Implementations

- Epetra (Essential Petra):
  - Current production version.
  - Restricted to real, double precision arithmetic.
  - Uses stable core subset of C++ (circa 2000).
  - Interfaces accessible to C and Fortran users.

- Tpetra (Templated Petra):
  - Next generation C++ version.
  - Templated scalar and ordinal fields.
  - Uses namespaces, and STL: Improved usability/efficiency.
- Jpetra (Java Petra):
  - Pure Java. Portable to any JVM.
  - Interfaces to Java versions of MPI, LAPACK and BLAS via interfaces.

Developers: Chris Baker, Mike Heroux, Rob Hoekstra, Alan Williams

Sandia National Laboratories

EpetraExt: Extensions to Epetra

- Library of useful classes not needed by everyone
- Most classes are types of “transforms”.
- Examples:
  - Graph/matrix view extraction.
  - Epetra/Zoltan interface.
  - Explicit sparse transpose.
  - Singleton removal filter, static condensation filter.
  - Overlapped graph constructor, graph colorings.
  - Permutations.
  - Sparse matrix-matrix multiply.
  - Matlab, MatrixMarket I/O functions.
  - Wrapper for PETSc aij matrices.
- Most classes are small, useful, but non-trivial to write.

Developers: Robert Hoekstra, Alan Williams, Mike Heroux, many others

Trilinos / PETSc Interoperability

- Epetra_PETScAIJMatrix class
  - Derives from Epetra_RowMatrix
  - Wrapper for serial/parallel PETSc aij matrices
  - Utilizes callbacks for matrix-vector product, getrow
  - No deep copies
- Enables PETSc application to construct and call virtually any Trilinos preconditioner
  - ML, Ifpack, AztecOO, …
  - All Trilinos options immediately available via parameter lists
- ML accepts fully constructed PETSc KSP solvers as smoothers
  - Fine grid only
  - Assumes fine grid matrix is really PETSc aij matrix
  - Complements Epetra_PETScAIJMatrix class
    - For any smoother with getrow kernel, PETSc implementation should be *much* faster than Trilinos
    - For any smoother with matrix-vector product kernel, PETSc and Trilinos implementations should be comparable

Thyra

- High-performance, abstract interfaces for linear algebra

- Offers flexibility through abstractions to algorithm developers
- Linear solvers (Direct, Iterative, Preconditioners)
  - Abstraction of basic vector/matrix operations (dot, axpy, mv).
  - Can use any concrete linear algebra library (Epetra, PETSc, BLAS).
- Nonlinear solvers (Newton, etc.)
  - Abstraction of linear solve (solve $Ax=b$).
  - Can use any concrete linear solver library:
    - AztecOO, Belos, ML, PETSc, LAPACK
- Transient/DAE solvers (implicit)
  - Abstraction of nonlinear solve.
  - … and so on.

Developers: Roscoe Bartlett, Kevin Long

Sandia National Laboratories

Stratimikos

- Stratimikos created Greek words "stratigiki“ (strategy) and "grammikos“ (linear)
- Defines class Thyra::DefaultLinearSolverBuilder.
- Provides common access to:
    - Linear Solvers: Amesos, AztecOO, Belos, …
    - Preconditioners: Ifpack, ML, …
  - Reads in options through a parameter list (read from XML?)
  - Accepts any linear system objects that provide
    - Epetra_Operator / Epetra_RowMatrix view of the matrix
    - SPMD vector views for the RHS and LHS (e.g. Epetra_[Multi]Vector objects)
- Provides uniform access to linear solver options that can be leveraged across multiple applications and algorithms

Key Points

- Stratimikos is an important building block for creating more sophisticated linear solver capabilities!

Stratimikos Parameter List and Sublists

```
<ParameterList name="Stratimikos">
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <Parameter name="Preconditioner Type" type="string" value="Ifpack"/>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="Amesos">
      <Parameter name="Solver Type" type="string" value="Klu"/>
      <ParameterList name="Amesos Settings">
        <Parameter name="MatrixProperty" type="string" value="general"/>
        ...
        <ParameterList name="Mumps"> ... </ParameterList>
        <ParameterList name="Superludist"> ... </ParameterList>
      </ParameterList>
    </ParameterList>
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <Parameter name="Max Iterations" type="int" value="400"/>
        <Parameter name="Tolerance" type="double" value="1e-06"/>
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          ...
        </ParameterList>
      </ParameterList>
      ...
    </ParameterList>
    <ParameterList name="Belos"> ... </ParameterList>
  </ParameterList>
<ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="0"/>
      <ParameterList name="Ifpack Settings">
        <Parameter name="fact: level-of-fill" type="int" value="0"/>
        ...
      </ParameterList>
    </ParameterList>
    <ParameterList name="ML"> ... </ParameterList>
  </ParameterList>
</ParameterList>
```

Top level parameters

Linear Solvers

Preconditioners

Sublists passed on to package code!

Every parameter and sublist is handled by Thyra code and is fully validated!

"Skins"

- PyTrilinos provides Python access to Trilinos packages
  - Uses SWIG to generate bindings.
  - Epetra, AztecOO, IFPACK, ML, NOX, LOCA, Amesos and NewPackage are supported.

Developer: Bill Spotz

- WebTrilinos: Web interface to Trilinos
  - Generate test problems or read from file.
  - Generate C++ or Python code fragments and click-run.
  - Hand modify code fragments and re-run.
  - Will use during hands-on.

Developers: Ray Tuminaro, Jonathan Hu, and Marzio Sala

Sandia National Laboratories

Whirlwind Tour of Packages

Discretizations **Methods** **Core** **Solvers/Preconditioners**

Amesos

- Interface to direct solvers for distributed sparse linear systems (KLU, UMFPACK, SuperLU, MUMPS, ScaLAPACK)

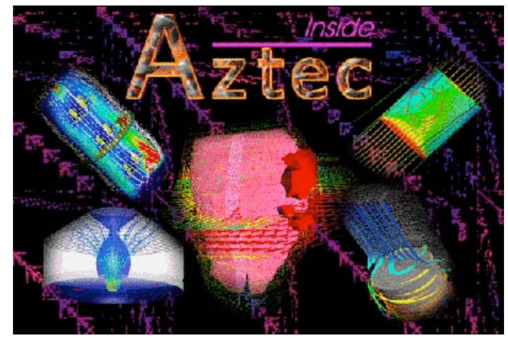
- Challenges:
  - No single solver dominates
  - Different interfaces and data formats, serial and parallel
  - Interface often changes between revisions
- Amesos offers:
  - A single, clear, consistent interface, to various packages
  - Common look-and-feel for all classes
  - Separation from specific solver details
  - Use serial and distributed solvers; Amesos takes care of data redistribution
  - Native solvers: KLU and Paraklete

Developers: Ken Stanley, Marzio Sala, Tim Davis

Sandia National Laboratories

AztecOO

- Krylov subspace solvers: CG, GMRES, Bi-CGSTAB,…
- Incomplete factorization preconditioners



- Aztec is the workhorse solver at Sandia:
  - Extracted from the MPSalsa reacting flow code.
  - Installed in dozens of Sandia apps.
  - 1900+ external licenses.
- AztecOO improves on Aztec by:
  - Using Epetra objects for defining matrix and RHS.
  - Providing more preconditioners/scalings.
  - Using C++ class design to enable more sophisticated use.
- AztecOO interfaces allows:
  - Continued use of Aztec for functionality.
  - Introduction of new solver capabilities outside of Aztec.

Developers: Mike Heroux, Alan Williams, Ray Tuminaro

Belos

- Next-generation linear solver library, written in templated C++.

- Provide a generic framework for developing iterative algorithms for solving large-scale, linear problems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - **Operator-vector products:** `Belos::MultiVecTraits`, `Belos::OperatorTraits`
  - **Orthogonalization:** `Belos::OrthoManager`, `Belos::MatOrthoManager`
  - **Status tests:** `Belos::StatusTest`, `Belos::StatusTestResNorm`
  - **Iteration kernels:** `Belos::Iteration`
  - **Linear solver managers:** `Belos::SolverManager`
- AztecOO provides solvers for $Ax=b$, what about solvers for:
  - Simultaneously solved systems w/ multiple-RHS: $AX = B$
  - Sequentially solved systems w/ multiple-RHS: $AX_i = B_i$, $i=1,\ldots,t$
  - Sequences of multiple-RHS systems: $A_iX_i = B_i$, $i=1,\ldots,t$
- Many advanced methods for these types of linear systems
  - Block methods: block GMRES [Vital], block CG/BICG [O'Leary]
  - "Seed" solvers: hybrid GMRES [Nachtigal, et al.]
  - Recycling solvers: recycled Krylov methods [Parks, et al.]
  - Restarting techniques, orthogonalization techniques, …

Developers: Heidi Thornquist, Mike Heroux, Mike Parks,
Rich Lehoucq, Teri Barth

Sandia National Laboratories

IFPACK: Algebraic Preconditioners

- Overlapping Schwarz preconditioners with incomplete factorizations, block relaxations, block direct solves.

- Accept user matrix via abstract matrix interface (Epetra versions).
- Uses Epetra for basic matrix/vector calculations.
- Supports simple perturbation stabilizations and condition estimation.
- Separates graph construction from factorization, improves performance substantially.
- Compatible with AztecOO, ML, Amesos. Can be used by NOX and ML.

Developers: Marzio Sala, Mike Heroux

Sandia National Laboratories

ML: Multi-level Preconditioners

- Smoothed aggregation multigrid, domain decomposition preconditioning, nonsymm. multigrid

- Critical technology for scalable performance of some key apps.
- ML compatible with other Trilinos packages:
  - Accepts user data as Epetra_RowMatrix object (abstract interface). Any implementation of Epetra_RowMatrix works.
  - Implements the Epetra_Operator interface. Allows ML preconditioners to be used with AztecOO, Belos, Anasazi.
- Can also be used independently of other Trilinos packages.

Developers: Ray Tuminaro, Jonathan Hu, Chris Siefert, Michael Gee

Sandia National Laboratories

Anasazi

- Next-generation eigensolver library, written in templated C++.

- Provide a generic framework for developing iterative algorithms for solving large-scale eigenproblems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - **Operator-vector products:** `Anasazi::MultiVecTraits`, `Anasazi::OperatorTraits`
  - **Orthogonalization:** `Anasazi::OrthoManager`, `Anasazi::MatOrthoManager`
  - **Status tests:** `Anasazi::StatusTest`, `Anasazi::StatusTestResNorm`
  - **Iteration kernels:** `Anasazi::Eigensolver`
  - **Eigensolver managers:** `Anasazi::SolverManager`
  - **Eigenproblem:** `Anasazi::Eigenproblem`
  - **Sort managers:** `Anasazi::SortManager`
- Currently has solver managers for three eigensolvers:
  - Block Krylov-Schur
  - Block Davidson
  - LOBPCG
- Can solve:
  - standard and generalized eigenproblems
  - Hermitian and non-Hermitian eigenproblems
  - real or complex-valued eigenproblems

Developers: Heidi Thornquist, Mike Heroux, Chris Baker, Rich Lehoucq, Ulrich Hetmaniuk

Sandia National Laboratories

NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

Broyden's Method

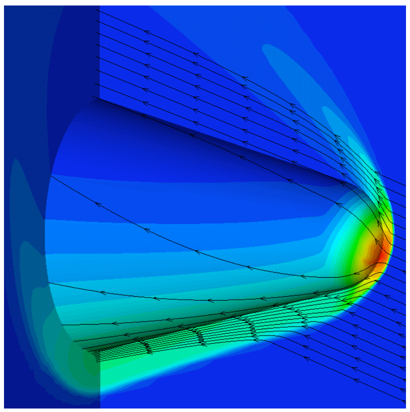
$$M_B = f(x_c) + B_c d$$

Newton's Method

$$M_N = f(x_c) + J_c d$$

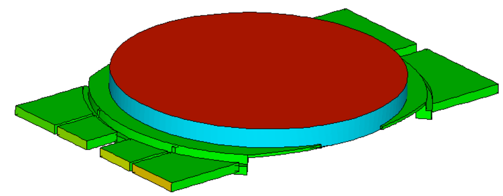
Tensor Method

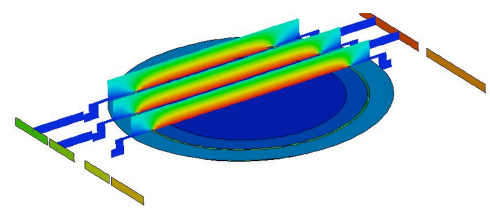
$$M_T = f(x_c) + J_c d + \frac{1}{2} T_c d d$$

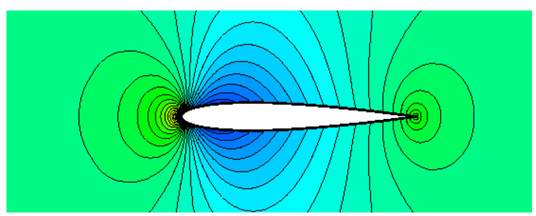


Globalizations

Line Search	Trust Region
Interval Halving	Dogleg
Quadratic	Inexact Dogleg
Cubic	
More'-Thuente	







Jacobian Estimation
- Graph Coloring
- Finite Difference
- Jacobian-Free Newton-Krylov

http://trilinos.sandia.gov/packages/nox

Implementation
- Parallel
- OO-C++
- Independent of the linear algebra package!

Developers: Tammy Kolda, Roger Pawlowski

LOCA

- Library of continuation algorithms

- Provides
  - Zero order continuation
  - First order continuation
  - Arc length continuation
  - Multi-parameter continuation (via Henderson's MF Library)
  - Turning point continuation
  - Pitchfork bifurcation continuation
  - Hopf bifurcation continuation
  - Phase transition continuation
  - Eigenvalue approximation (via ARPACK or Anasazi)

Developers: Andy Salinger, Eric Phipps

Sandia National Laboratories

MOOCHO & Aristos

- MOOCHO: Multifunctional Object-Oriented arCHitecture for Optimization
  - Large-scale invasive simultaneous analysis and design (SAND) using reduced space SQP methods.

Developer: Roscoe Bartlett

- Aristos: Optimization of large-scale design spaces
  - Invasive optimization approach based on full-space SQP methods.
  - Efficiently manages inexactness in the inner linear system solves.

Developer: Denis Ridzal

Sandia National Laboratories

Full Vertical Solver Coverage

Optimization **Unconstrained:** **Constrained:**	Find $u \in \Re^n$ that minimizes $g(u)$ Find $x \in \Re^m$ and $u \in \Re^n$ that minimizes $g(x,u)$ s.t. $f(x,u) = 0$	**MOOCHO**
Bifurcation Analysis	Given nonlinear operator $F(x,u) \in \Re^{n+m}$ - For $F(x,u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$ s	**LOCA**
Transient Problems **DAEs/ODEs:**	Solve $f(\dot{x}(t), x(t), t) = 0$ $t \in [0,T], x(0) = x_0, \dot{x}(0) = x_0'$ for $x(t) \in \Re^n, t \in [0,T]$	**Rythmos**
Nonlinear Problems	Given nonlinear operator $F(x) \in \Re^m \rightarrow \Re$ Solve $F(x) = 0 \quad x \in \Re^n$	**NOX**
Linear Problems **Linear Equations:** **Eigen Problems:**	Given Linear Ops (Matrices) $A, B \in \Re^{m \times n}$ Solve $Ax = b$ for $x \in \Re^n$ Solve $A\nu = \lambda B\nu$ for (all) $\nu \in \Re^n, \quad \lambda \in$	**AztecOO** **Belos** **Ifpack, ML, etc...** **Anasazi**
Distributed Linear Algebra **Matrix/Graph Equations:** **Vector Problems:**	Compute $y = Ax; A = A(G); A \in \Re^{m \times n}, G \in \Im^{m \times n}$ Compute $y = \alpha x + \beta w; \alpha = \langle x, y \rangle; x, y \in \Re^n$	**Epetra** **Tpetra**

Sensitivities
(Automatic Differentiation: Sacado)

Trilinos Integration into an Application

Where to start?

http://trilinos.sandia.gov

Export Makefile System

Once Trilinos is built, how do you link against the application?

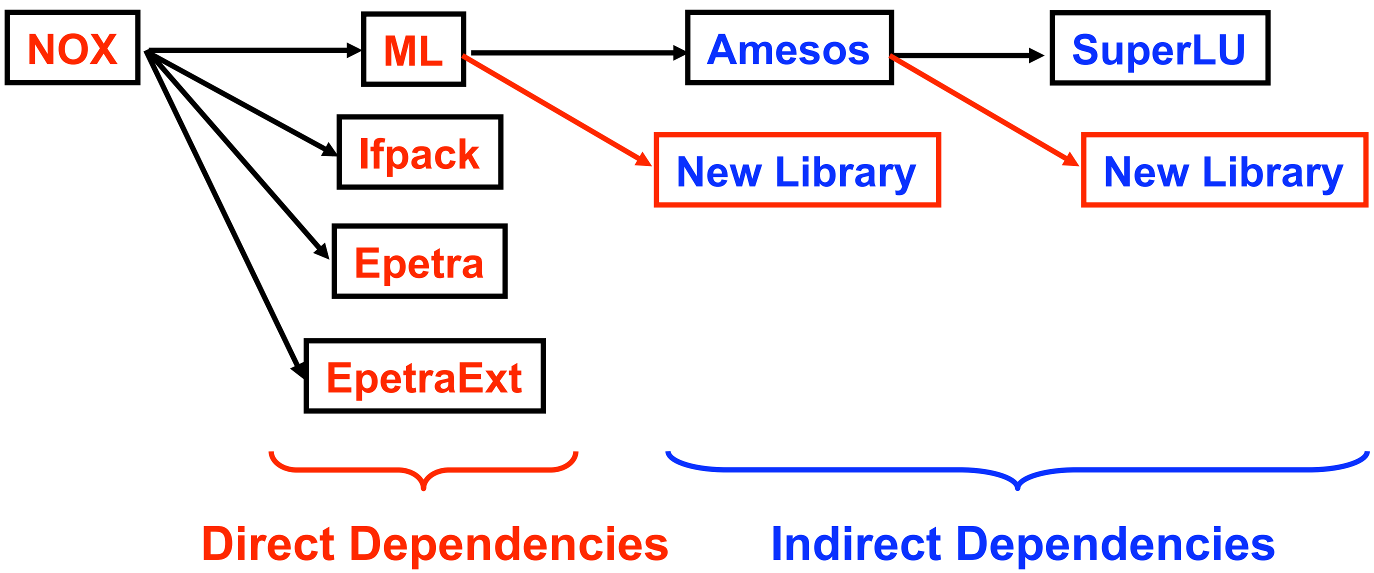
There are a number of issues:

- Library link order:
    - -lnoxepetra -lnox –lepetra –lteuchos –lblas –llapack
- Consistent compilers:
    - g++, mpiCC, icc…
- Consistent build options and package defines:
    - g++ -g –O3 –D HAVE_MPI –D _STL_CHECKED

Answer: Export Makefile system

Why Export Makefiles are Important

- The number of packages in Trilinos has exploded.
- As package dependencies (especially optional ones) are introduced, more maintenance is required by the top-level packages:



NOX either must:

- Account for the new libraries in its configure script (not scalable)
- Depend on direct dependency packages to supply them through "export" Makefiles.

Export Makefiles in Action

```
########################################################################
## Example Makefile for a user application that does not use autoconf
## - Uses lapack concrete instantions for group and vector
## - Must use gnu-make (gmake) if the "shell" command is invoked
########################################################################

##
## Set the Trilinos install directory
##
TRILINOS_INSTALL_DIR = /home/rppawlo/trilinos-local-install

##
## Include any direct Trilinos library dependencies - in this case only nox
##
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.nox.macros
include $(TRILINOS_INSTALL_DIR)/include/Makefile.export.nox

##
## Use one of the following lines (2nd line is for non-gnumake platforms)
##
COMPILE_FLAGS = $(shell perl $(TRILINOS_INSTALL_DIR)/include/strip_dup_incl_paths.pl $(NOX_CXXFLAGS) $(NOX_DEFS)
$(NOX_CPPFLAGS) $(NOX_INCLUDES))

COMPILE_FLAGS = $(NOX_CXXFLAGS) $(NOX_DEFS) $(NOX_CPPFLAGS) $(NOX_INCLUDES)

##
## Use one of the following lines (2nd line is for non-gnumake platforms)
##
LINK_FLAGS = $(shell perl $(TRILINOS_INSTALL_DIR)/include/strip_dup_libs.pl $(NOX_LIBS))
LINK_FLAGS = $(NOX_LIBS)

## ## Build your application code ##
main.exe: main.o
	$(NOX_CXXLD) $(NOX_CXXFLAGS) -o main.exe main.o $(LINK_FLAGS)

main.o: main.cpp
	$(NOX_CXX) $(COMPILE_FLAGS) -c main.cpp

clean: rm -f *.o main.exe *~
```

Concluding Remarks

Trilinos Availability / Information

- Trilinos and related packages are available via LGPL.
  - Current release (9.0) is “click release”. Unlimited availability.
  - Trilinos alpha release (cmake build preview): July, 2009
  - Trilinos Release 10.0: September 2009.
- Trilinos Awards:
  - 2004 R&D 100 Award.
  - SC2004 HPC Software Challenge Award.
  - Sandia Team Employee Recognition Award.
  - Lockheed-Martin Nova Award Nominee.
- More information:
  - http://trilinos.sandia.gov
- 6th Annual Trilinos User Group Meeting in October 2008 @ SNL
  - talks available for download
- Next TUG is November 3-5, 2009 at Sandia/Albuquerque

Useful Links

Trilinos website: http://trilinos.sandia.gov

Trilinos tutorial: http://trilinos.sandia.gov/Trilinos8.0Tutorial.pdf

Trilinos mailing lists: http://trilinos.sandia.gov/mail_lists.html

Trilinos User Group (TUG) meetings:

http://trilinos.sandia.gov/events/trilinos_user_group_2008

http://trilinos.sandia.gov/events/trilinos_user_group_2007